重新认识npm

#### 序言

对于npm，很多开发者都只停留在“用”，不太了解它具体的工作流程。

本文主要是简单说明npm的工作流程，学习如何更好地管理它，并且使用它开发一些工具。

**注意：本文中的「包」、「项目」、「依赖」是同一个东西。**

#### 用途

npm（全称：Node Package Manager），是JavaScript运行时环境Node.js的默认包管理器。

npm将开发代码和工具包连接在一起，在node环境中引入包，node会自动在对应开发环境中的 **node\_modules** 文件夹找到对应的包。

**node\_modules**由npm生成，npm通过声明的配置文件（package.json）来安装对应的包到**node\_modules**中**。**

值得注意的是，即使配置文件中没有对应的包名，直接将包复制到**node\_modules**，也是可以直接引入的。

#### package.json

package.json 不只是npm配置文件，也可以是整个工程项目的基础。

大部分的前端开发工程，都是基于node进行的，使用node就离不开npm。

先来看下package.json的内容

{

"name": "sasai-admin", // 工程名/包名

"version": "0.1.0", // 版本号

"private": true, // 是否私有包；私有包发布到npm无法被搜索到，特定用户才能用

"author": "vue project",

"main": "bin/index.js"

"types": "bin/index.d.ts",

"bin": {

"vue-cli-service": "bin/vue-cli-service.js"

},

"description": "vue project",

"license":"",

"scripts": { // 脚本

"serve": "vue-cli-service serve",

"lint": "vue-cli-service lint",

},

"dependencies": { // 生产依赖

"axios": "^0.26.0",

},

"devDependencies": { // 开发依赖

"pre-commit": "^1.2.2",

"@vue/cli-service": "~5.0.0",

},

"browserslist": [ // 浏览器兼容列表（不一定会有）

"> 1%",

"last 2 versions",

"not dead"

],

"engines": { // 包版本管理

"node": ">=12.0.0",

},

"pre-commit": [

"lint"

]

}

##### 必填属性

* name是项目（包）的名称；
* version 版本号；
* description是描述信息
* author是作者相关信息（发布时用到）；
* license是开源协议（发布时用到）；

##### private属性

private属性记录当前的包是否私有的；

当为true时，无法发布到npm

##### main属性

设置入口。

npm在引入包文件时，会根据这个字段去查找文件暴露的模块。

举个例子：

const axios = require('axios')

npm会自动去 **axios** 的 **package.json** 的 **main** 找到入口，再引入入口文件的模块。



可以从图中看到，axios的入口文件是index.js，所以在项目中引入的时候，实际上就是引入这个index.js。

##### bin属性

**可执行命令**：在bin中声明一些命令及对应的入口，就可以在当前环境或者全局环境中（取决于安装方式）运行该入口，

比如vue-cli，它在bin中声明了vue-cli-service，并且指向了 bin/vue-cli-service.js，当你安装了vue-cli，就可以在命令行工具中，通过执行vue-cli-service来运行vue-cli-service.js。

**bin可以声明多个**



##### types属性

TypeScript的类型信息入口，如果包支持TypeScript，编译完成后，也需要声明types属性，暴露类型信息的入口。



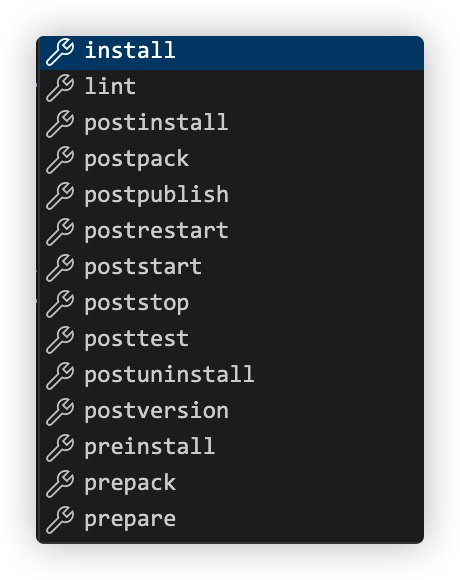
##### scripts

scripts属性用于配置一些脚本命令，以键值对的形式存在；

配置后我们可以通过执行命令来运行脚本：

npm run ${key}

除了自己配置的脚本，npm还内置了一些声明周期系列的钩子函数：



比如install，意思是在完成 npm install事件后，执行事件；

还有prepublish，这是在发布包之前执行事件，这个在发布环节中用的比较多。

##### dependencies属性

dependencies属性是指定无论开发环境还是生产环境都需要依赖的包；

比如vue、axios等等

##### devDependencies属性

开发环境依赖包，比如：webpack、vue-cli

我们会通过 以下命令，将它安装到devDependencies属性中：

npm install webpack --save-dev

##### 依赖版本号

我们会发现安装的依赖版本出现：^2.0.3或~2.0.3，这是什么意思呢？

这里简单解释下 ^ 和 ~ 的区别：

^x.y.z：表示x是保持不变的，y和z永远安装最新的版本；

~x.y.z：表示x和y保持不变的，z永远安装最新的版本；

npm的版本号管理遵循Semver版本规范，具体请查阅：[语义化版本 2.0.0 | Semantic Versioning (semver.org)](https://semver.org/lang/zh-CN/)

##### engines属性

engines属性用于指定Node和NPM的版本号；

在安装的过程中，会先检查对应的引擎版本，如果不符合就会报错；

事实上也可以指定所在的操作系统 "os" : [ "darwin", "linux" ]，只是很少用到。

##### pre-commit

这是 git 检查的一个工具，意思是在commit执行一些scripts中的脚本命令。

一般情况下，会执行一遍规范检查。

它需要先安装「pre-commit」这个依赖。

// 执行scripts中的lint命令

"pre-commit": [

"lint",

]

#### npm run 做了什么？

在了解npm run ${key} 做了什么之前，我们要先明白**系统环境**、**全局环境**和**局部环境**这三个概念**。**

**系统环境：**指的是在系统任何目录下，都可以运行的命令。比如node和java，我们在安装完成后，打开终端，在任意位置都能通过命令行执行它们。

**全局环境，**和系统环境类似，同样在系统任意目录下都可执行，不同的是，它依赖于npm，需要借助来运行。

**但不需要使用npm命令来调用。**

可以使用npm来安装一些全局环境的依赖：

// 安装

npm install @vue/cli-service -g

// 使用

vue-cli-service server

**局部环境**：作用于当前或者某个项目下的命令，它只能在执行项目工程下执行。

比如webpack（当然它也可以是全局的），在某个项目中使用npm安装，它就可以在当前项目使用。

**它无法直接调用，需要借助npm命令**。

直接在项目下安装对应依赖即可：

// 安装

npm install @vue/cli-service --save-dev

// 声明:package.json

{

"script": {

"sever": "vue-cli-service server"

}

}

// 使用

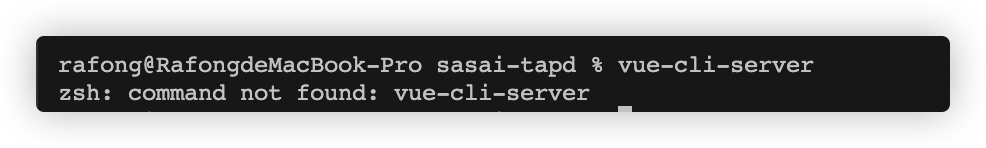
npm run sever

那么，我们想了解下npm run 到底做了什么。

npm run serve // vue-cli-service serve

这行命令做了什么？

它只是执行了vue-cli-service这个命令，那么我们直接执行它会怎样：



报错，告诉我们并不存在。

它无法直接调用，**因为我们没有在全局环境变量中声明vue-cli-server的入口。**

而npm帮我们找到了。

怎么做到的？

当我们安装vue-cli的时候，npm会检查vue-cli的配置文件是否存在 **bin** ，

如果存在，会在node\_modules会创建一个隐藏目录：.bin ，存放该可执行命令入口。

当我们执行npm run serve时，npm会从下到上分别找到**局部环境**、**全局环境**和**系统环境**的可执行命令，当找到任意一个就中断查找，并且立刻执行：

1. npm会先在 node\_modules/.bin 中找可执行命令，如果找到，立刻执行，找不到就往上查找
2. 在npm全局依赖中的bin找可执行命令，如果找到，立刻执行，找不到再往上查找
3. 在系统环境中找可执行命令，找到就执行，找不到，报错。

总结一下：

1. 每个npm包可以声明可执行命令，在安装的时候npm会将可执行命令进行集中管理，便于查找
2. 在工程项目中执行命令，npm会依次从下至上（局部环境、全局环境和系统环境）查找可执行命令，优先执行。
3. 全局环境也可以在当前系统用户的任意目录下执行

#### 开始一个工程

在不使用脚手架工具情况下，搭建一个新的工程。

1. 新建一个目录，执行npm命令创建package.json

npm init #手动模式

npm init -y #自动填写默认信息

然后就可以定义项目结构和安装依赖了。

#### 开发npm包

搭建一个工程后，可以尝试开发一个npm包。

开发npm包跟普通的业务开发差不多，但有几个点需要注意：

1. 必须声明main或者bin
2. 包名需要唯一，或者使用私有包名
3. 忽略开发目录和非必要代码

##### 声明入口

作为一个工具包，必须清楚定义自己的入口，才可以进行引入或者调用。

**main和bin是可以共存的。**

根据自己工具的定位进行定义：

1. 生产代码用的组件或者插件，定义main；
2. 开发工具可以定义一个或者多个bin，调用可执行命令执行。
3. 如果是TypeScript，需要声明types类型信息

##### 定义包名

作为一个需要发布的包，包名至关重要：它必须是唯一的。

不能与npm上现存的其它包名相同。

// package.json

{

"name": "sasai-publish" // 唯一

}

当有其它包与你定义的包名冲突时，也是使用私有包名：在包名前加上你的用户名或者团队名称：

{

"name": "@sasaiweb/sasai-publish"

}

这样就可以婉转解决唯一性的问题。

**注意：**

* **如果使用团队名称，必须先加入团队**
* **包名用户名必须和发布者（或者团队）相同。**

##### 忽略非必要文件

开发完成发布之前，需要对自己的包进行梳理，哪些是需要发布，哪些是不需要发布上去的。

一般情况下，开发目录是不需要发布上去的，还有一些涉及**非公开的配置（账号密码等）。**

在工程目录下，新建一个.npmignore文件，把一些不需要发布的文件和目录加进去即可：

node\_modules

src

.babelrc

webpack.config.js

package-lock.json

.DS\_Store

bin/.DS\_Store

#### 发布npm包

##### 账号注册

发布前需要在npm上进行账号注册：[npm (npmjs.com)](https://www.npmjs.com/)

注册完成后请将账号发给，加入sasaiweb的团队，后续可以发布团队包。

##### 登录

在当前工程目录下登录npm账号：

npm login

根据提示依次输入账号、密码、邮箱以及邮箱验证码，操作完成后即成功。

##### 发布

npm上有两种包：**公开包和私有包**。

* 公开包是免费的，发布用户无法付费，但所有人都可以看到并使用
* 私有包是付费的，一般情况下，在包名加入用户名（团队名）默认就是私有包。

所以，在发布的时候需要区分不同的情况。

1、如果是个人包发布的话，直接使用：

npm publish

即可发布成功

2、如果加入了私有包名（用户名或者团队），而且非付费用户，需要声明发布公开包：

npm publish --access public

1. 如果是私有包名，且已付费，就直接发布。

##### 版本更新

每次进行重新发布，都需要更新版本号，相同版本号不可以重复发布，但可以进行**强制替换**和**发布撤销（**这里不展开**）**。

在工程目录下，输入以下命令，即可更新版本号。

npm version patch

npm发布常见问题：

[发布npm包填坑小记 | 思忆技术 (si-yee.com)](https://blog.si-yee.com/2020/09/21/%E5%8F%91%E5%B8%83npm%E5%8C%85%E5%A1%AB%E5%9D%91%E5%B0%8F%E8%AE%B0/)

参考文档：

[Node.js——npm你不知道的细节 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/390274102)

[NPM包（模块）发布、更新、撤销发布 - 掘金 (juejin.cn)](https://juejin.cn/post/6844903769365282829)